Fast Approximation to Spherical Harmonic Rotation

Jaroslav Křivánek, Jaakko Konttinen, Sumanta N. Pattanaik, Kadi Bouatouch

▶ **To cite this version:**

Jaroslav Křivánek, Jaakko Konttinen, Sumanta N. Pattanaik, Kadi Bouatouch. Fast Approximation to Spherical Harmonic Rotation. [Research Report] PI 1728, 2005, pp.14. inria-00000169

HAL Id: inria-00000169
https://inria.hal.science/inria-00000169

Submitted on 19 Jul 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1728

FAST APPROXIMATION TO SPHERICAL HARMONIC ROTATION

JAROSLAV KŘIVÁNEK, JAAKKO KONTTINEN, SUMANTA PATTANAIK,
KADI BOUATOUCH

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Fast Approximation to Spherical Harmonic Rotation

Jaroslav Křivánek* , Jaakko Konttinen** , Sumanta Pattanaik*** , Kadi Bouatouch****

Systèmes numriques
Projet Siames

Publication interne n ° 1728 — Juin 2005 — 14 pages

Abstract: Rotating functions represented by spherical harmonics is an important part of many real-time lighting and global illumination algorithms. For some of them a per-vertex or even per-pixel rotation is required, which implies the necessity of an efficient rotation procedure. The speed of any of the existing rotation procedures is, however, not able to meet the requirements of real-time lighting or fast global illumination. We present an efficient approximation of the spherical harmonic rotation applicable for small rotation angles. We replace the general spherical harmonic rotation matrix by its first or second order Taylor expansion, which significantly decreases the computation involved in the rotation. Our approximation decreases the asymptotic complexity of the rotation — the higher the order of spherical harmonics, the higher the speed-up. We apply the proposed approximation in global illumination and real-time lighting by environment maps.

Key-words: spherical harmonics rotation, spherical harmonics lighting, radiance caching, environment mapping, normal mapping

(Résumé : tsvp)

* jkrivane@irisa.fr
** jaakko@cs.ucf.edu
*** sumant@cs.ucf.edu
**** kadi@irisa.fr

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes
Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Approximation rapide de la rotation d'harmoniques sphériques

Résumé : La rotation de fonctions représentées par des harmoniques sphériques est une opération importante dans les algorithmes de calcul d'éclairage en temps réel et de simulation d'éclairage global. Pour certains algorithmes, une rotation est effectuée pour chaque somment ou même chaque pixel. Cette procédure doit donc être efficace en terme de temps de calcul. Malheureusement les procédures existantes ne permettent pas de répondre aux exigences des algorithmes de calcul d'éclairage parce qu'elles sont coûteuses en terme de nombre d'opérations. Dans ce rapport, nous proposons une approximation efficace de la rotation d'harmoniques sphériques, cette approximation étant applicable dans le cas de petits angles de rotation. Nous remplaçons la matrice générale de rotation d'harmoniques sphériques par son développement en série de Taylor au premier ou au seconde ordre, ce qui réduit la complexité de calcul de façon significative. Cette approximation réduit aussi la complexité asymptotique de la rotation. En effet, plus l'ordre des harmoniques sphériques sera élevé, plus l'approximation est rapide. Nous appliquons l'approximation proposée au cas de l'éclairage global et à celui du calcul en temps réel de l'éclairage à l'aide de cartes d'environnement.

Mots clés : rotation d'harmoniques sphériques, éclairage à l'aide d'harmoniques sphériques, cache de luminance, carte d'environnement, carte de normales

1 Introduction

When using spherical basis functions (e.g. spherical harmonics or wavelets) for real-time shading with environment lighting [KSS02, SKS02, LK03, SHHS03, WTL04, LSSS04, NRH04], one has to face the problem of aligning the environment lighting (represented in the global coordinate frame) with the reflectance function, or BRDF (represented in the local coordinate frame at each surface point). The alignment is achieved through a rotation of a (hemi)spherical function. Kautz et al. [KSS02] perform the rotation of the environment lighting represented by spherical harmonics for each vertex during real-time rendering and report that the rotation is the bottleneck. Ng et al. [NRH04] avoid the rotation by storing the BRDF multiple times, pre-rotated to the global frame for different surface normal directions. This approach wastes memory, prohibits the use of high frequency BRDFs and does not allow anisotropic BRDFs. Precomputed radiance transfer [SKS02, LK03, SHHS03, WTL04, LSSS04] avoids the rotation problem for smooth surfaces since the alignment is included in the transfer matrix stored per-vertex. However, in the case of normal mapped surfaces, a per-pixel rotation is needed even for precomputed radiance transfer. Also for global illumination computation, efficiency of hemispherical function rotation can be critical [KGPB05].

To our knowledge, no simple rotation procedure exists for wavelet representation. Functions represented by spherical harmonics can be rotated by a linear transformation of the representation coefficients [Gre03], but the existing procedures [IR96, IR98, CIGR99, KSS02] are too slow for per-pixel rotation in real-time.

In this paper we address the rotation of functions represented by spherical harmonics. We propose an efficient approximation of the spherical harmonic rotation based on replacing the general spherical harmonic rotation matrix with its Taylor expansion. We show that our approximation has lower computational complexity in terms of spherical harmonic order than the previous methods. Our method is also faster and we show that a rotation can be performed in real-time on a per-pixel basis. We apply the proposed rotation in global illumination and real-time lighting with environment maps. The ability to perform the SH rotation per-pixel allows us to decouple the illumination quality from the number of vertices. We demonstrate this by rendering normal mapped objects illuminated by environment maps in real-time.

The next section provides the background on spherical harmonic rotation, Section 3 describes our rotation approximation. Applications and results are presented in Section 4 and Section 5 concludes the work.

2 Background

2.1 Spherical Harmonics

Any spherical function $L(\omega)$ can be approximated in terms of spherical harmonics as $L(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m Y_l^m(\omega)$, where ω is a direction in 3D, Y_l^m are the spherical harmonics (abbreviated SH) [Gre03] and n is the SH approximation *order*. Coefficients λ_l^m constitute the representation of $L(\omega)$ with respect to the SH basis. There are n^2 coefficients in the approximation of order n. Spherical harmonics of equal l index form a *band*, with one harmonic in the first band ($m = 0$), three in the second band ($m = -1, 0, 1$), five in the third band ($m = -2, -1, 0, 1, 2$), etc. Although the coefficients have two indices l and m, they are stored in a one dimensional array $[\lambda_0^0, \lambda_1^{-1}, \lambda_1^0, \lambda_1^1, \ldots]$, indexed by $i = l(l+1) + m$. This layout is used in the example code in this paper.

2.2 Spherical Harmonic Rotation

Problem Statement. Given a vector of SH coefficients $\Lambda = \{\lambda_l^m\}$ representing a spherical function $L(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m Y_l^m(\omega)$, find a vector of coefficients $\Upsilon = \{\upsilon_l^m\}$ representing the rotated function $L(\mathscr{R}^{-1}(\omega)) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \upsilon_l^m Y_l^m(\omega)$, where $\mathscr{R}$ is the desired rotation.

Rotation of any function represented by SH of order n can be exactly represented by SH of order n. The rotation can be carried out as a linear transformation $\Upsilon = \mathbf{R}\Lambda$ with a block-sparse rotation matrix $\mathbf{R}$ (Figure 1). Note that coefficients between different SH bands do not interact. The problem is how to construct $\mathbf{R}$ for a desired 3D rotation and order n. Different ways of solving this task are described in [Gre03]. Our approach to SH rotation, described in Section 3, avoids explicit construction of $\mathbf{R}$. We compare our approach with the methods of Ivanic and Ruedenberg [IR96, IR98] and the ZXZXZ decomposition of Kautz et al. [KSS02].

Ivanic and Ruedenberg [IR96, IR98] construct $\mathbf{R}$ recurrently, starting from $\mathbf{R}^1$ continuing over $\mathbf{R}^2$ up to $\mathbf{R}^l$ for any given l. Elements of the block $\mathbf{R}^l$ are computed from elements of $\mathbf{R}^{l-1}$ and $\mathbf{R}^1$ using rules summarized in [IR98, Gre03]. The procedure is relatively efficient, but too slow to be used for each pixel or even each vertex in real-time.

A more efficient SH rotation can be achieved with the method of [KSS02] that we call here the ZXZXZ decomposition. A general 3D rotation is first decomposed into ZYZ Euler angles (α, β, γ). The rotation around Y by angle β is then expressed as a rotation around X by $\pi/2$, a general rotation around Z by β and a rotation around X by $-\pi/2$. The angle of the two rotations around X is fixed, therefore the rotation matrices for them can be pre-computed. The number of non

$$\mathbf{R} = \left[\begin{array}{c|ccc|ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \dots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right] = \left[\begin{array}{cccc} 1 & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{R}^1 & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{array}\right]$$

Figure 1: Form of the SH rotation matrix. (After [Green 2003]).

zero elements in those matrices is only a fourth of that of a general SH rotation matrix and a general rotation around Z is very simple (see Appendix C), thus the efficiency of this method.

Direct3D API [Mic04] provides the `D3DXSHRotate()` call that rotates a function represented by spherical harmonics. The implementation is probably based on explicit formulas for the elements of the rotation matrix in terms of Euler angles [SKS02], since it only works for orders up to $n = 6$. It is slower than the method of Ivanic and Ruedenberg [IR96, IR98].

Choi et al.'s method [CIGR99] performs the rotation in complex space and then converts the results back to real space (our spherical harmonics and coefficient vectors are *real*). According to [Gre04] this procedure is slower than the method of Ivanic and Ruedenberg [IR96, IR98]. The source code for these two methods is available online [Wil04].

None of the listed methods is fast enough to allow per-pixel rotation, which inspired us to develop our fast rotation approximation.

3 Our Contribution: Fast Rotation Approximation

This section describes our fast approximation of the SH rotation using the Taylor expansion of the rotation matrix. We show that the approximation decreases the rotation complexity from $O(n^3)$ to $O(n^2)$, where n is the order.

According to Euler's rotation theorem, any rotation may be described using three angles. We decompose rotations using the ZYZ convention and express them as three subsequent rotations around Z, Y and Z axes by angles α, β and γ, respectively, i.e. $\mathbf{R} = \mathbf{R}_Z(\alpha)\mathbf{R}_Y(\beta)\mathbf{R}_Z(\gamma)$.

The rotation around Z is simple and efficient (Appendix C). It remains to find the rotation matrix $\mathbf{R}_Y(\beta)$. Our main contribution consists in replacing this matrix by its Taylor expansion at $\beta = 0$:

$$\mathbf{R}_Y(\beta) \approx \mathbf{I} + \beta \frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0) + \frac{\beta^2}{2}\frac{\mathrm{d}^2\mathbf{R}_Y}{\mathrm{d}\beta^2}(0).$$

where $\mathbf{I}$ is the identity matrix. Computation of the derivative matrices is described in Appendix A. The first derivative matrix $\frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0)$ has nonzero elements only on the super- and subdiagonals and the second derivative matrix $\frac{\mathrm{d}^2\mathbf{R}_Y}{\mathrm{d}\beta^2}(0)$ has non-zeros only on the main diagonal and on the diagonal just below the subdiagonal and just above the superdiagonal (Figure 2). Therefore, the resulting rotation matrix approximation is very sparse. The rotation matrix $\mathbf{R}_Y(\beta)$ does not have to be explicitly constructed at all because we know where the nonzero elements are.

In practice we use a "1.5-th order" Taylor expansion, where any non-diagonal elements of the second derivative matrix are ignored. The C code below shows how the Y rotation is carried out using the "1.5-th order" Taylor expansion.

$$\frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\mathrm{d}\mathbf{R}^1_Y}{\mathrm{d}\beta}(0) & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}\mathbf{R}^2_Y}{\mathrm{d}\beta}(0) & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}\mathbf{R}^3_Y}{\mathrm{d}\beta}(0) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \qquad \frac{\mathrm{d}^2\mathbf{R}_Y}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}^1_Y}{\mathrm{d}\beta^2}(0) & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}^2_Y}{\mathrm{d}\beta^2}(0) & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}^3_Y}{\mathrm{d}\beta^2}(0) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where

$$\frac{\mathrm{d}\mathbf{R}^1_Y}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\frac{\mathrm{d}\mathbf{R}^2_Y}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.73 & 0 \\ 0 & 0 & 1.73 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\frac{\mathrm{d}\mathbf{R}^3_Y}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 1.22 & 0 & 0 & 0 & 0 & 0 \\ -1.22 & 0 & 1.58 & 0 & 0 & 0 & 0 \\ 0 & -1.58 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2.45 & 0 & 0 \\ 0 & 0 & 0 & 2.45 & 0 & -1.58 & 0 \\ 0 & 0 & 0 & 0 & 1.58 & 0 & -1.22 \\ 0 & 0 & 0 & 0 & 0 & 1.22 & 0 \end{pmatrix}$$

where

$$\frac{\mathrm{d}^2\mathbf{R}^1_Y}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\frac{\mathrm{d}^2\mathbf{R}^2_Y}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 1.73 \\ 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 1.73 & 0 & -1 \end{pmatrix}$$

$$\frac{\mathrm{d}^2\mathbf{R}^3_Y}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} -1.5 & 0 & 1.94 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 1.94 & 0 & -2.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 3.87 & 0 \\ 0 & 0 & 0 & 0 & -8.5 & 0 & 1.94 \\ 0 & 0 & 0 & 3.87 & 0 & -4 & 0 \\ 0 & 0 & 0 & 0 & 1.94 & 0 & -1.5 \end{pmatrix}$$

Figure 2: First (left) and second (right) derivative of the Y rotation matrix at $\beta = 0$. (Numbers are rounded to three significant digits.)

```
/** Rotate around Y using the 1.5-th order Taylor expansion
  @param beta        angle of rotation around Y
 */
void shRotYdiff15(int order, float* dest, const float* src,
                  const float* dySubDiag, const float* ddyDiag,
                  float beta) {
 float bbeta = 0.5f*beta*beta;
 dest[0] = src[0];
 for(int i=1; i<order*order-1; i++) {
  dest[i] = src[i] * (1.0f + bbeta*ddyDiag[i]) +
   beta * (dySubDiag[i]*src[i-1] - dySubDiag[i+1]*src[i+1]);
 }
 dest[i] = src[i] * (1.0f + bbeta*ddyDiag[i]) +
           beta * dySubDiag[i] * src[i-1];
}
```

The arrays `dySubDiag` and `ddyDiag` contain the subdiagonal of $\frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0)$ and the diagonal of $\frac{\mathrm{d}^2\mathbf{R}_Y}{\mathrm{d}\beta^2}(0)$ respectively. They are computed just once at the start-up of the application and remain constant throughout the run-time. The superdiagonal of $\frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0)$ does not have to be stored, since the first derivative matrix is, like any other infinitesimal rotation matrix, antisymmetric [Wei04].

All components for the full rotation $\mathbf{R} = \mathbf{R}_Z(\alpha)\mathbf{R}_Y(\beta)\mathbf{R}_Z(\gamma)$ are now available. The rotation proceeds as follows:

1. Decompose rotation into the ZYZ Euler angles α, β and γ.
2. Rotate around Z by α (see Appendix C) .
3. Use `shRotYdiff15()` to rotate around Y by β.
4. Rotate around Z by γ.

It has to be emphasized that the described procedure only *approximates* the rotation and is usable only if the angle of rotation around Y is small. An application using our approximation has to assure that this condition holds. Section 3.2 compares the approximation error for the first, "1.5-th" and second order Taylor expansions.

3.1 Complexity

We compare the complexity of Ivanic and Ruedenberg's rotation [IR96, IR98] and rotation by ZXZXZ decomposition [KSS02] with the complexity of our approximation. The complexities are expressed in terms of order n.

Ivanic and Ruedenberg's method. The number of non-zero elements in a general SH rotation matrix (Figure 1) for order n is $N_{nz}(n) = \sum_{i=1}^{n}(2i-1)^2 = n(4n^2-1)/3$. Computation of each element of the matrix using Ivanic and

Ruedenberg's method [IR96, IR98] is a constant-time operation, therefore the complexity of the SH rotation matrix construction is $O(n^3)$. Complexity of transforming a SH coefficient vector with the matrix is also $O(n^3)$.

ZXZXZ Decomposition. One Z rotation involve $N_Z(n) = 2n(n-1)$ multiplications; the cost of one X rotation is $N_X(n) = \sum_{i=1}^{n}(i^2 - i + 1) = n(n^2+2)/3$. Rotation of one SH vector with the ZXZXZ decomposition thus costs $3N_Z(n) + 2N_X(n) = n(2n^2 + 18n - 14)/3 \in O(n^3)$ multiplications. This is only about a half of the number of multiplications needed for transforming a vector by a full SH rotation matrix $\mathbf{R}$ and there is no explicit construction of the matrix.

Our rotation approximation. There are $N_{dY}(n) = 5n^2$ multiplications in `rotYdiff15()`. The total cost of our rotation $2N_Z(n) + N_{dY}(n) = 9n^2 - 4n \in O(n^2)$ is asymptotically lower than the previous methods. The advantage of our method in terms of speed becomes more pronounced as the order n increases, the downside being the lower accuracy for higher n.

3.2 Error Analysis.

Let $\mathbf{R}_Y(\beta)$ be the correct matrix for rotation around Y by β and let $\mathbf{R}'_Y(\beta)$ be our approximation. For a given coefficient vector Λ, the approximation error $E(\beta)$ is given by the L_2 norm

$$E(\beta) = \|\mathbf{R}_Y(\beta)\Lambda - \mathbf{R}'_Y(\beta)\Lambda\| = \|(\mathbf{R}_Y(\beta) - \mathbf{R}'_Y(\beta))\Lambda\| = \|\mathbf{D}(\beta)\Lambda\|$$

Maximum of $E(\beta)$ over all unit length Λ is the L_2 norm of the matrix $\mathbf{D}(\beta)$, which is equal to the greatest singular value of $\mathbf{D}(\beta)$. Average $E(\beta)$ over all unit length Λ is the average singular value of $\mathbf{D}(\beta)$. Figure 3 shows the maximum and average error $E(\beta)$ and also the actual measured $E(\beta)$ for a Phong lobe $\cos^7(\theta)$. Although the maximum error grows very quickly with β, the results for the Phong lobe show good accuracy up to $\beta = 25°$.

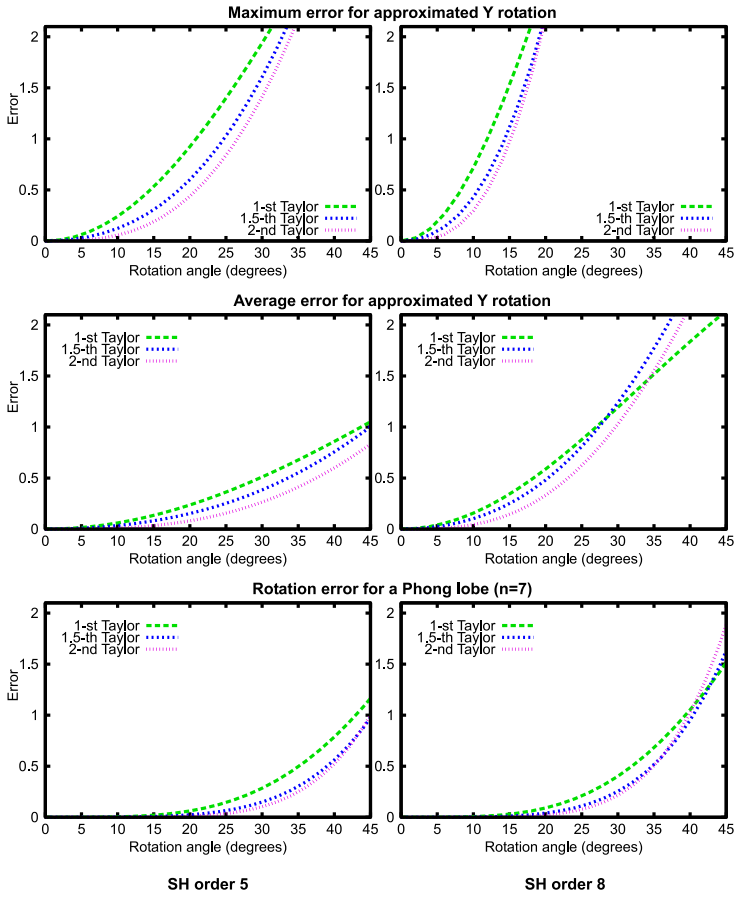

Figure 3: Approximation error $E(\beta)$ as a function of β for spherical harmonics of order 5 (left) and 8 (right). The plots in the first row show the maximum error for any unit length vector, the second row shows the average error over all unit length vectors and the third row shows the actual error for a Phong lobe $\cos^7(\theta)$. Error is expressed as the Euclidean distance between the coefficient vectors. Each plot shows the error for the 1st-, 1.5th-, and 2nd order Taylor expansion.

4 Applications and Results

4.1 Real-time Environment Lighting

In this section we extend the technique of Kautz et al. [KSS02] with normal mapping through the use of our rotation approximation. A brief review of the original technique is as follows. Kautz et al. use spherical harmonics to perform real-time shading of surfaces with arbitrary BRDFs due to low-frequency environment lighting. The BRDF is represented as a 2D table, whose each element stores the SH coefficients of the BRDF for one fixed outgoing (viewing) direction. Environment lighting is also represented by spherical harmonics. The lighting integral for a given viewing direction is computed as a dot product of lighting coefficients and BRDF coefficients for that outgoing direction. In this technique, the variation of incident lighting due to surface orientation is limited by the number of vertices in the mesh. The rendering algorithm proceeds as follows:

1. [Per-vertex, CPU] Rotate the lighting coefficients to the local coordinate frame of vertex v_p. Send the rotated coefficient as vertex data to the GPU.
2. [Per-pixel, GPU] Look up the BRDF coefficients for the viewing direction transformed to the local coordinate frame.
3. [Per-pixel, GPU] Compute the dot product of local lighting and BRDF coefficients.

Our extension decouples the lighting computation from the number of vertices and allows modulating the surface normal on a per-pixel basis by a normal map. We achieve this by modifying the original rendering algorithm in the following way (new steps are in italics):

1. [Per-vertex, CPU] Rotate the lighting coefficients to the local coordinate frame of vertex v_p. Send the rotated coefficient as vertex data to the GPU.
2. [Per-pixel, GPU] *Look up the normal map (normal map represents the modulation of the local coordinate frame at the pixel with respect to the frame given by the interpolated per-vertex normals).*
3. [Per-pixel, GPU] Look up the BRDF coefficients for the viewing direction transformed to the *modulated* local coordinate frame.
4. [Per-pixel, GPU] *Use our SH rotation approximation to rotate the BRDF coefficients from the modulated local frame to the interpolated per-vertex local frame.*
5. [Per-pixel, GPU] Compute the dot product of local lighting and BRDF coefficients.

The modulation of the surface normal by the normal map is usually limited to rather small angles; we can therefore safely use our rotation approximation. Moreover, thanks to the approximation simplicity, we were able to implement the per-pixel rotation in a pixel shader of the graphics hardware.

The above extension leads to a significant improvement of visual quality as illustrated in Figures 4 and 5. It also allows using meshes with lower number of vertices than the original technique, which improves the overall rendering performance. Similar approach can also be used to augment visual richness of the spherical harmonics-based precomputed radiance transfer techniques [Sloan 2002, Kautz 2002], by applying the per-pixel rotation on the transferred radiance.

To simplify the normal mapping one can ignore the per-pixel rotation (step 4) and use the normal map only to modulate the local frame for the BRDF look-up. Unlike our method, this simplified normal mapping generates flat looking surfaces and it also does not capture color variations on the surface bumps that stem from the modulated surface normal.

Results. Figures 4 and 5 compare our results with the simplified normal mapping. We used spherical harmonics of order n=5 (25 coefficients). The rotation approximation used the "1.5-th order" Taylor expansion for bands l=1 to l=3 and the first order Taylor expansion for band l=4. Due to the limited pixel shader instruction count we had to use four passes to accommodate 25 coefficients. The frame rates for these images at resolution 800×600 are:

	Simplified	Ours
Vase (891 vertices)	46 fps	58 fps
Sphere (560 vertices)	54 fps	65 fps
Plane (25 vertices)	56 fps	75 fps

These figures were measured on a 2.26GHz Pentium IV PC with ATI Radeon 9800 Pro GPU. The drop in the frame rate due to the rotation is more pronounced for the very low-frequency mesh, where the rendering time is determined mostly by fragment processing.

Figure 4: Detail of a normal mapped vase rendered with our SH rotation (right) and with the simplified normal mapping (left). Normal mapping with our SH rotation is more successful at conveying the shape approximated by the normal map. The vase is illuminated by the St Peter's Basilica environment map; the BRDF comes from a measurement of a brushed metal [Wes].

4.2 Radiance Caching

Here we briefly review the radiance caching algorithm and describe how we have applied our rotation approximation. Radiance caching [KGPB05] is a generalization of Ward et al.'s [WRC88] irradiance caching algorithm. It accelerates indirect illumination computation on glossy surfaces by sparsely sampling, caching and interpolating the incoming radiance. The incoming radiance at a point is a hemispherical function represented by spherical or hemispherical harmonics [GKPB04] coefficients stored in the radiance cache. For a ray hitting a glossy surface at a point $\mathbf{p}$, radiance cache is queried. If no cached radiance record is found near $\mathbf{p}$, hemisphere above $\mathbf{p}$ is sampled and the incoming radiance coefficients are stored in the radiance cache. If cached records are found near $\mathbf{p}$, the incoming radiance coefficients are interpolated with the formula:

$$\Lambda(\mathbf{p}) = \frac{\sum_S (\mathbf{R}_i \Lambda_i) w_i(\mathbf{p})}{\sum_S w_i(\mathbf{p})} \tag{1}$$

No translation gradients are used here. Weight of the i-th cache record with respect to point $\mathbf{p}$ is given by $w_i(\mathbf{p}) = (\|\mathbf{p}-\mathbf{p}_i\|/R_i + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i})^{-1}$, where $\mathbf{p}_i$ is the position of the i-th cache record, $\mathbf{n}_i$ is the normal at $\mathbf{p}_i$, R_i is the harmonic mean length of rays sampling the hemisphere above $\mathbf{p}_i$, $\mathbf{n}$ is the normal at $\mathbf{p}$, $S = \{i | w_i(\mathbf{p}) > 1/a\}$ and a is a user defined allowed error. The important thing here is the rotation $\mathbf{R}_i$ that has to be used to align the coordinate frames at $\mathbf{p}_i$ and $\mathbf{p}$: the cached incoming radiance *has to be rotated* before the interpolation is possible (Figure 6). This means that there is one or more rotation for each interpolation (each pixel on a visible glossy surface).

Due to the interpolation criterion the normals at $\mathbf{p}_i$ and $\mathbf{p}$ are always similar (if they were not, $\mathbf{p}_i$ would not be used for interpolation at $\mathbf{p}$). Angle of rotation around the Y axis in the Euler ZYZ decomposition of $\mathbf{R}_i$ corresponds to the angle between normals $\mathbf{n}_i$ and $\mathbf{n}$ and hence it is always small. We can therefore safely use our rotation approximation.

We use the approximated rotation for $\beta < \beta_{\text{lim}}$ and the more costly ZXZXZ decomposition otherwise. If we kept β_{lim} constant, increasing the radiance caching error a would lead to more frequent use of the ZXZXZ decomposition for rotation and the interpolation would slow down. This is certainly not what the user expects from increasing allowed caching error. To rectify this, we set the limiting angle to $\beta_{\text{lim}} = 1.25a$ (derivation in Appendix B), which allows more error in rotation if the user allows more error in interpolation. As a consequence, the percentage of ZXZXZ rotations is constant regardless of a in a given scene, which is what the user expects. In this setting, our approximation and the ZXZXZ rotation may not meet in a visually continuous fashion at β_{lim} if a is high ($a > 0.3$ in our scenes). Is such case, however, the caching artifacts are more pronounced and the rotation artifacts go unnoticed.

Results. For results in this section we used the "1.5-th Taylor expansion" of the Y rotation matrix (all non diagonal elements of the second derivative matrix are ignored). When using our rotation approximation, we set the limiting angle β_{lim} so that it is never exceeded.

In Figure 7 we compare the results of radiance caching obtained by the correct and the approximated rotation. Instead of a side-by-side comparison, in which the results are visually indistinguishable, we show a color coded difference between the two methods. Image areas exhibiting the maximum error are usually very curved, and the artifacts, if any, are well masked.

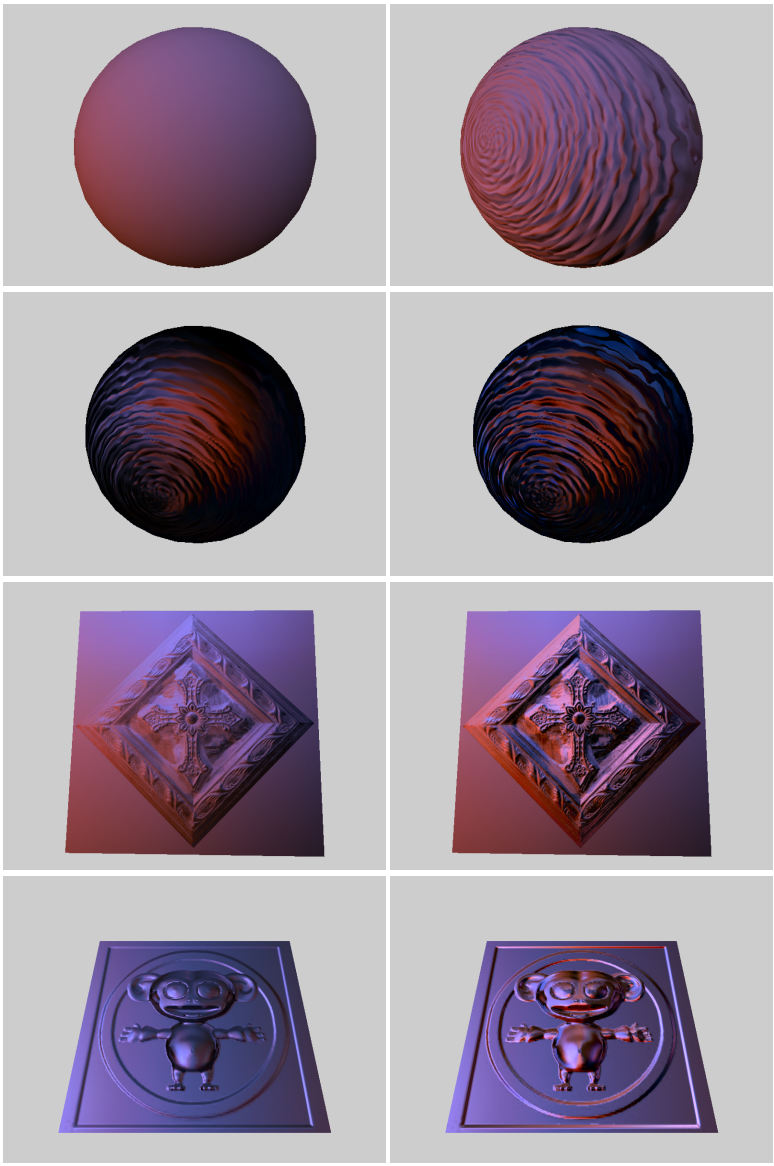

Figure 5: More results of the normal mapping with our SH rotation (right) compared to the simplified normal mapping (left). The BRDFs used were (from top to bottom) Lambertian, Phong, Ward isotropic, Ward anisotropic. The objects are illuminated by the Grace Cathedral environment map. Note the color variations on the surface bumps captured by our method. For the Lambertian surface (top), the simplified normal mapping does not work since the BRDF is view independent. Our method is also more successful at revealing the effects of BRDF anisotropy (bottom).

Table 1 shows the rendering times for the flamingo and the sphere (Figure 7 left) with radiance caching. The rendering time only includes the interpolation from the cache. For SH order $n = 6$, our method is 4 times faster than the ZXZXZ decomposition; for $n = 10$ it is 6 times faster.

5 Conclusion

We presented a fast technique for rotating functions represented by spherical harmonics. We approximated the spherical harmonic rotation matrix by its Taylor expansion which increases its sparsity. Our technique decreases the rotation complexity and is faster than previous rotation algorithms. Although our rotation approximation is accurate only for small rotation angles, we have demonstrated its practical usefulness in real-time and off-line rendering. The rotation algorithm is simple enough to fit in the pixel shader of standard graphics hardware, which allows to apply the rotation on a per-

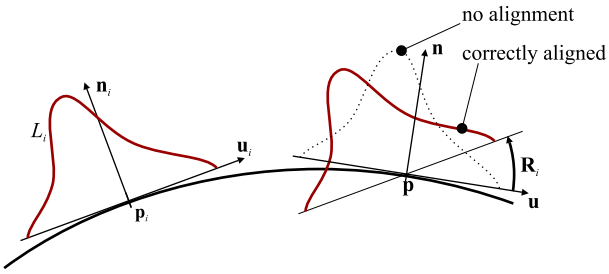

Figure 6: Rotation $\mathbf{R}_i$ aligns the coordinate frame at $\mathbf{p}_i$ and $\mathbf{p}$ before interpolation is possible in radiance caching. (After [Křivánek et al. 2004].)

Order	6		10	
	RT	TPR	RT	TPR
Flamingo				
Ignore	10.3 s	—	11.2 s	—
Our method	**12.8 s**	**0.68 μs**	**16.9 s**	**1.54 μs**
ZXZXZ	21.2 s	2.96 μs	47.4 s	9.83 μs
Ivanic	47.3 s	10.1 μs	192 s	49.1 μs
DirectX	76.4 s	17.9 μs	—	—
Sphere				
Ignore	3.30 s	—	3.96 s	—
Our method	**4.28 s**	**0.65 μs**	**6.13 s**	**1.44 μs**
ZXZXZ	7.08 s	2.51 μs	16.8 s	8.57 μs
Ivanic	17.8 s	9.63 μs	75.8 s	47.8 μs
DirectX	30.3 s	17.9 μs	—	—

Table 1: Rendering times for the flamingo and sphere images (Figure 7 left) with radiance caching. The rendering time only includes interpolation from the cache. Various rotation methods are used for interpolation: Ignore (rotation is ignored), our method, ZXZXZ decomposition, the method of Ivanic and Ruedenberg and the DirectX rotation. 'RT' is the frame rendering time and 'TPR' is the time per rotation. There were $1,226,917 \times 3 = 3,680,751$ rotations for the flamingo and $501,420 \times 3 = 1,504,260$ rotations for the sphere.

pixel basis in real-time. We demonstrated this by shading normal mapped surfaces with arbitrary BRDFs by environment lighting. We have also applied our rotation approximation in global illumination computation on glossy surfaces.

In future work, we would like to extend our rotation-based normal mapping to precomputed radiance transfer. We also believe that a similar approach can be used to approximate rotation of functions represented by wavelets.

References

[CIGR99] Cheol Ho Choi, Joseph Ivanic, Mark S. Gordon, and Klaus Ruedenberg. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *J. Chem. Phys.*, 111(19):8825–8831, 1999.

[GKPB04] Pascal Gautron, Jaroslav Křivánek, Sumanta N. Pattanaik, and Kadi Bouatouch. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*, pages 321–330, June 2004.

[Gre03] Robin Green. Spherical harmonic lighting: The gritty details. In *Game Developpers' Conference*, 2003.

[Gre04] Robin Green. Personal communication, 2004.

[IR96] Joseph Ivanic and Klaus Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *J. Phys. Chem.*, 100(15):6342–6347, 1996.

[IR98] Joseph Ivanic and Klaus Ruedenberg. Additions and corrections : Rotation matrices for real spherical harmonics. *J. Phys. Chem. A*, 102(45):9099–9100, 1998.

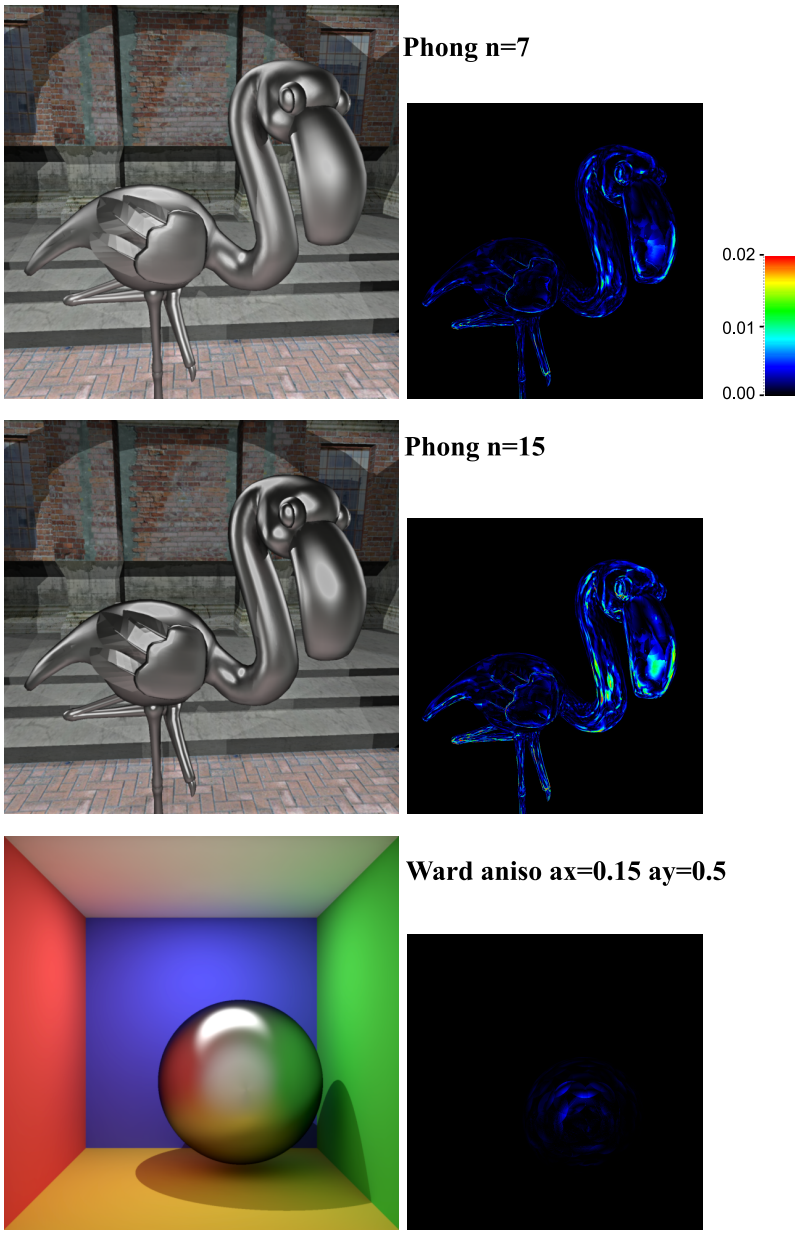

Figure 7: **Left.** Radiance caching renderings obtained with approximated rotation. The flamingo is assigned a Phong BRDF with the exponent of 7 (top) or 15 (middle) and the sphere has an anisotropic Ward BRDF [Ward 1992] with $\alpha_x = 0.15$, $\alpha_y = 0.5$. SH order $n = 10$ is used for all renderings. **Right.** Color coded difference between images with approximated and correct rotation, measured on a $[0,1]$ RGB scale. The difference is below the visual threshold of 1% for most pixels.

[KGPB05] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics (Accepted for publication)*, 2005.

[KSS02] Jan Kautz, Peter-Pike Sloan, and John Snyder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 291–296. Eurographics Association, 2002.

[LK03] Jaakko Lehtinen and Jan Kautz. Matrix radiance transfer. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 59–64. ACM Press, 2003.

[LSSS04] Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of the Eurographics Symposium on Rendering*, pages 337–344,

2004.

[Mic04] Microsoft. Directx 9.0 SDK update, October 2004.

[NRH04] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph.*, 23(3):477–487, 2004.

[SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.*, 22(3):382–391, 2003.

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536. ACM Press, 2002.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH*, pages 265–272. ACM Press, 1992.

[Wei04] Eric W. Weisstein. Infinitesimal rotation. From *MathWorld*–A Wolfram Web Resource. http://mathworld.wolfram.com/InfinitesimalRotation.html, 2004.

[Wes] Stephen H. Westin. Lafortune BRDF for RenderMan. http://www.graphics.cornell.edu/ westin/lafortune/lafortune.html.

[Wil04] Don Williamson. Spherical harmonic rotation. Available from http://www.donw.co.uk/, 2004.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH '88*, pages 85–92, 1988.

[WTL04] Rui Wang, John Tran, and David Luebke. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proceedings of the Eurographics Symposium on Rendering*, pages 345–354, 2004.

A SH Rotation Matrix Derivative

Here we describe the computation of the k-th derivative matrix $\frac{\mathrm{d}^k \mathbf{R}_Y}{\mathrm{d}\beta^k}$. The algorithm is based on Ivanic and Ruedenberg's rotation matrix construction [IR96, IR98] and retains its structure. Elements of the derivative matrix block $\frac{\mathrm{d}^k \mathbf{R}^l_Y}{\mathrm{d}\beta^k}$ are indexed by m_1 and m_2 and we denote them $R^{(k)}_Y(l,m_1,m_2)$. We start with bands $l=0$ and $l=1$:

$$
\begin{array}{llll}
R^{(k)}_Y(0,\ 0,\ 0) = 1^{(k)} & R^{(k)}_Y(1,\ 0,\ 0) = \cos^{(k)}(\beta) \\
R^{(k)}_Y(1,\ -1,\ -1) = 1^{(k)} & R^{(k)}_Y(1,\ 0,\ 1) = -\sin^{(k)}(\beta) \\
R^{(k)}_Y(1,\ -1,\ 0) = 0 & R^{(k)}_Y(1,\ 1,\ -1) = 0 \\
R^{(k)}_Y(1,\ -1,\ 1) = 0 & R^{(k)}_Y(1,\ 1,\ 0) = \sin^{(k)}(\beta) \\
R^{(k)}_Y(1,\ 0,\ -1) = 0 & R^{(k)}_Y(1,\ 1,\ 1) = \cos^{(k)}(\beta)
\end{array}
$$

where $\beta = 0$ and $1^{(k)}$ is the derivative of one ($1^{(k)} = 1$ if $k = 0$ and $1^{(k)} = 0$ if $k > 0$). For higher bands, we compute simultaneously the zero-th, first, second, ... *maxderiv*-th derivative:

for $l = 2 \ldots n-1$ **do**
  for $k = 0 \ldots maxderiv$ **do**
    for $m_1 = -l \ldots l$ **do**
      for $m_2 = -l \ldots l$ **do**

$$
\begin{aligned}
R^{(k)}(l,m_1,m_2) \quad := \quad & u^l_{m_1 m_2} \cdot dU^{(k)}(l,m_1,m_2) + \\
& v^l_{m_1 m_2} \cdot dV^{(k)}(l,m_1,m_2) + \\
& w^l_{m_1 m_2} \cdot dW^{(k)}(l,m_1,m_2)
\end{aligned}
$$

      end for
    end for
  end for
end for

Numerical coefficients $u^l_{m_1m_2}$, $v^l_{m_1m_2}$ and $w^l_{m_1m_2}$ are the same as in the original paper [IR98]. Functions dU, dV and dW are defined as:

$$dU^{(k)}(l,m_1,m_2) = dP^{(k)}(l,m_1,m_2,0)$$

$$dV^{(k)}(l,m_1,m_2) = \begin{cases} dP^{(k)}(l,m_1-1,m_2,1) - \\ dP^{(k)}(l,-m_1+1,m_2,-1) & \text{if } m_1 > 1 \\ \hline \sqrt{2}dP^{(k)}(l,0,m_2,1) & \text{if } m_1 = 1 \\ \hline dP^{(k)}(l,1,m_2,1) + \\ dP^{(k)}(l,-1,m_2,-1) & \text{if } m_1 = 0 \\ \hline \sqrt{2}dP^{(k)}(l,0,m_2,-1) & \text{if } m_1 = -1 \\ \hline dP^{(k)}(l,-m_1-1,m_2,-1) + \\ dP^{(k)}(l,m_1+1,m_2,1) & \text{if } m_1 < -1 \end{cases}$$

$$dW^{(k)}(l,m_1,m_2) = \begin{cases} dP^{(k)}(l,m_1+1,m_2,1) + \\ dP^{(k)}(l,-m_1-1,m_2,-1) & \text{if } m_1 > 0 \\ \hline dP^{(k)}(l,m_1-1,m_2,1) - \\ dP^{(k)}(l,-m_1+1,m_2,-1) & \text{otherwise} \end{cases}$$

with

$$dP^{(k)}(l,m_1,m_2,i) = \begin{cases} dT^{(k)}(1,i,0,l-1,m_1,m_2) & \text{if } |m_2| < l \\ \hline dT^{(k)}(1,i,1,l-1,m_1,l-1) - \\ dT^{(k)}(1,i,-1,l-1,m_1,-l+1) & \text{if } m_2 = l \\ \hline dT^{(k)}(1,i,1,l-1,m_1,-l+1) + \\ dT^{(k)}(1,i,-1,l-1,m_1,l-1) & \text{if } m_2 = -l \end{cases}$$

and

$$dT^{(k)}(l,m_1,m_2,l',m_1',m_2') = \sum_{i=0}^{k} \binom{k}{i} R_Y^{(k)}(l,m_1,m_2) \cdot R_Y^{(k-i)}(l',m_1',m_2').$$

Function dT implements the product derivative rule $(fg)^{(k)} = \sum_{i=0}^{k} \binom{k}{i} f^{(i)} g^{(k-i)}$, where $f^{(k)}$ denotes the k-th derivative.

B Rotation Approximation Limiting Angle

The decision whether a radiance cache record i will be included in the interpolated value at $\mathbf{p}$ depends on its weight as $a > 1/w_i(\mathbf{p})$ (a is the user defined allowed error). Weight is a function of the distance $\|\mathbf{p}-\mathbf{p}_i\|$ and the angle between normals $\beta = \angle(\mathbf{n},\mathbf{n}_i)$. Consider a constant curvature surface with the osculating circle radius r. Then $\|\mathbf{p}-\mathbf{p}_i\| = 2r\sin\frac{\beta}{2}$ and therefore the weight is only a function of β, i.e. $1/w_i(\mathbf{p}) = f(\beta) = \frac{2r}{R_i}\sin\frac{\beta}{2} + \sqrt{1-\cos\beta}$, where R_i is the harmonic mean distance. The aim is to find for a given a and fixed r the value of $\beta_{\lim}$ such that for all accepted radiance cache records, the normal divergence is never more than $\beta_{\lim}$. To this end we need the inverse function of f, which is impossible to find analytically. Instead we take the first order Taylor expansion of f at $\beta = 0$ which is $f(\beta) \approx \beta(1/\sqrt{2} + \frac{r}{R_i})$ and we find $\beta < a(1/\sqrt{2} + \frac{r}{R_i})^{-1}$. We choose $\frac{r}{R_i} = 0.1$ and we get $\beta_{\lim} = 1.25a$ (in degrees $\beta_{\lim} = 70.1a$). The choice $\frac{r}{R_i} = 0.1$ means that all surfaces of curvature $10/R_i$ or smaller will be rendered without exceeding $\beta_{\lim}$.

C SH Rotation around Z-axis

The Z-rotation is computed efficiently without constructing the rotation matrix $\mathbf{R}_Z(\alpha)$ using the following procedure:

for $l = 0 \ldots n-1$ **do**
  $v_l^0 := \lambda_l^0$
  for $m = 1 \ldots l$ **do**
    $v_l^{-m} := \lambda_l^{-m}\cos(m\alpha) - \lambda_l^m \sin(m\alpha)$
    $v_l^m \;\; := \lambda_l^{-m}\sin(m\alpha) + \lambda_l^m \cos(m\alpha)$
  end for
end for

The sines and cosine of multiple angles can be computed with the recurrence formula:

$$\begin{aligned}\sin(m\alpha) &= 2\sin((m-1)\alpha)\cos(\alpha) - \sin((m-2)\alpha)\\ \cos(m\alpha) &= 2\cos((m-1)\alpha)\cos(\alpha) - \cos((m-2)\alpha)\end{aligned}$$

The number of multiplications in the rotation procedure is $N_Z(n) = 2n(n-1)$ thus the complexity is $O(n^2)$.